Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 226 (2009) 39–48

www.elsevier.com/locate/entcs

Panoptes

An Exploration Tool for Formal Proofs

William M. Farmer[1,2] Orlin G. Grigorov[1,3]

Department of Computing and Software
McMaster University
Hamilton, Ontario, Canada

Abstract

Proof assistants aid the user in proving mathematical theorems by taking care of low-level reasoning details. Their user interfaces often present proof information as text, which becomes increasingly difficult to comprehend as it grows in size. Panoptes is a software tool that enables users to explore graphical representations of the formal proofs produced by the IMPS Interactive Mathematical Proof System. Panoptes automatically displays an IMPS deduction graph as a visual graph that can be easily manipulated by the user. Its facilities include target zooming, floating information boxes, node relabeling, and proper substructure collapsing.

Keywords: IMPS, deduction graph, proof tree, theorem prover, graph visualization, OCaml, OpenGL.

1 Introduction

A proof assistant is a software system for developing formal proofs. The user guides the development of an attempt to prove a conjecture, while many of the low-level details are done automatically by the proof assistant. Proof assistants are usually not equipped with sophisticated tools for exploring the "tree" of formulas that is produced by a proof attempt. However, the proof structure created in proving a conjecture can sometimes grow to a large size

[1] This research was supported by NSERC.

[2] Email: wmfarmer@mcmaster.ca

[3] Email: ogrigorov@gmail.com

doi:10.1016/j.entcs.2008.12.096

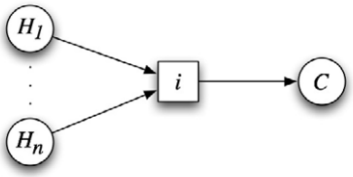
involving hundreds of formulas and inferences. In this case, the user can easily lose his or her way when exploring the proof and can miss seeing different parts of the proof with similar structure that could be merged if identified.

Panoptes, named after the all-seeing giant of Greek mythology, is a software system for exploring the proof structures produced by the IMPS Interactive Mathematical Proof System [3,4,5]. The proof structures that IMPS creates are certain kinds of graphs called deduction graphs. Panoptes automatically displays an IMPS deduction graph as a visual graph that can be manipulated by the user. Although Panoptes is designed to work with IMPS, it focuses on facilities that would be useful to many other proof assistants. This paper describes the facilities that Panoptes provides and gives an overview of its implementation.

2 Deduction Graphs in IMPS

A *deduction graph* [3] in IMPS is a directed bipartite graph used to represent a proof or proof attempt. It contains two types of nodes and arrows that connect a node of one type to a node of the other type. A *sequent node* represents a sequent consisting of a single formula called the *assertion* and a finite set of assumptions called the *context*. An *inference node* represents an inference from a finite set of sequents (the hypotheses) to a single sequent (the conclusion). An inference node has arrows pointing to it from the sequent nodes representing its hypotheses and an arrow pointing from it to a sequent node representing its conclusion. The *root node* of a deduction graph is a distinguished sequent node in the graph that represents the sequent to be proved.

For example, the figure



is a small deduction graph consisting of $n + 1$ sequent nodes and 1 inference node. This deduction graph represents the inference of the conclusion held by the sequent node C from the hypotheses held by the sequent nodes $H_1, \ldots, H_n$.

Since a sequent node can have more than one arrow into it, any number of alternate strategies can be represented in the deduction graph for proving a given sequent. Thus a deduction graph generally does not represent a single proof attempt, but rather a set of intertwined proof attempts. Deduction

graphs may contain cycles and may not be connected. A sequent node is said to be *grounded* if it is known to be valid. A deduction graph is a *proof* if its root node is grounded. A deduction graph that is a proof does not necessarily represent a proof tree; it may contain garbage, i.e., parts of the deduction graph that represent unneeded or unfinished alternate proof attempts.

3 Description of System

Panoptes serves as an add-on application, which runs concurrently with IMPS. It provides a graphical visualization of the deduction graph, which is synchronized with the internal representation of the deduction graph upon request by the user. The tool provides a large set of functions to ease the process of exploring the structure of the graph and understanding the logical development of the proof. The main design goal was to make graph manipulation easy, and the result is a program that provides an almost playful way of exploring the deduction graph. Another design goal was to make it easy to port to other theorem provers by encapsulating into a separate module the part that processes the input from the theorem prover.

3.1 Functionality

Apart from the graphical visualization of the deduction graph on the screen, the system provides a range of useful functionality to the user. The following are some of the major options available.

- **Target zooming.** The user can zoom in and out on parts of the graph by just pointing with the mouse and holding down a button. This is quite different from the standard way of zooming first and then scrolling to reach the point of interest, which can easily lead to confusion and disorientation of the user.
- **Collapsing.** Parts of the graph can be collapsed into special inference and sequent nodes. For instance, if the validity of a sequent node is reduced to the validity of one or more other sequent nodes through a number of proof steps, the user has the option to collapse all these steps into a special inference node, which consolidates the reasoning that reduces the goal to the subgoals. Similarly, cycles of equivalent sequent nodes can be collapsed into a special sequent node. Collapsing is very important when dealing with large deduction graphs since it enables secondary information to be hidden without compromising the semantic integrity of the deduction graph representation.
- **Labeling.** The user can freely label nodes and parts of the graph so that

these components can be identified with names that are more meaningful than the names generated by the system.

- **Floating information boxes.** Each node, regardless of its type, contains some information. In the case of a sequent node, this information comprises the sequent represented by the node. An inference node contains the inference rule that generated the represented inference, and a collapsed inference node contains the (possibly large) part of the graph that is hidden by the collapsing. Each node in the deduction graph has an information box that contains the information associated with it. These information boxes can be toggled between visible and nonvisible states. Additionally, when visible, an information box has a direct visible link to its associated node, which further enhances the efficiency of presenting the information to the user. Of course, these information boxes can be scaled, repositioned, and manipulated in many ways by merely pressing a button or dragging the mouse.
- **History of operations.** A comprehensive history of the operations applied to a deduction graph is kept at all times, so that the user can easily revert back to an earlier arrangement of the graph on the screen. Also, this function is important for preserving the effort invested into rearranging the graph between proof steps, which is possible due to the fact that IMPS only adds new nodes, but never removes nodes from the deduction graph.
- **Automatic layout and manual rearranging.** Upon startup, Panoptes provides an initial layout of the deduction graph. This allows Panoptes to fit the whole graph in the screen space provided by the system and also to minimize the crossing of edges as much as possible. In addition, the user is able to drag and drop components of the graph to either improve or modify the layout according to his or her preference, while the program automatically protects the connections (the arrows) between the nodes.

Additionally, appropriate automatic labeling and numbering of repetitions (in the case of inference nodes representing applications of the same inference rule) is automatically performed by the system. The power of color is also utilized: grounded nodes are colored in green, repeated nodes (i.e., nodes that complete a cycle or merge proof directions) in brown, collapsed inference nodes in purple, etc.

3.2 Implementation

A fully functional prototype of the proposed system has been developed in Objective Caml (OCaml) [9] using the LablGL library [6] that implements an interface to OpenGL [7] in OCaml.

The choice of OCaml as the programming language was made on the basis of its features, such as its support for modular design, as well as its automatic garbage collection system, type inferencing, and allowance for both imperative and functional programming styles. All of this adds up to a versatile language that is suitable for developing large projects with a reduced chance for programming errors and an increased runtime stability. Furthermore, OCaml is available for many operating systems including Linux and Mac OS X, which makes the tool portable to all systems that currently support IMPS.

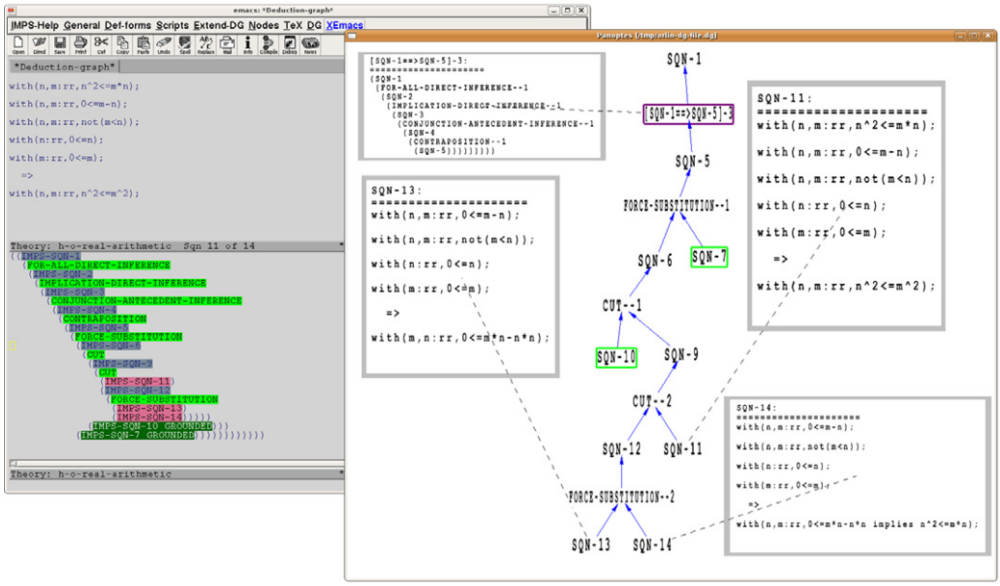
As for the graphical library, OpenGL is usually associated with three-dimensional graphical visualizations, but the system uses these capabilities for implementing different techniques. For instance, moving the graph or selected components of the graph closer or further away from the viewer creates the effect of zooming in contrast to the usual method of merely scaling the image. The advantage lies in OpenGL being a direct API to the 3D instruction set of the graphical hardware, and as such it provides a performance unmatched by the standard way of drawing graphics on the screen. The result is an application, which delegates all graphical computations and manipulations to the GPU, rather than to the CPU of the host machine, leaving the latter fully available for other work (such as that done by the IMPS reasoning engine). Consequently, a computer system equipped with a reasonably modern graphics card would be capable of running the prototype smoothly without burdening the user with unnecessary lags and delays.

3.3 Availability and Screenshots

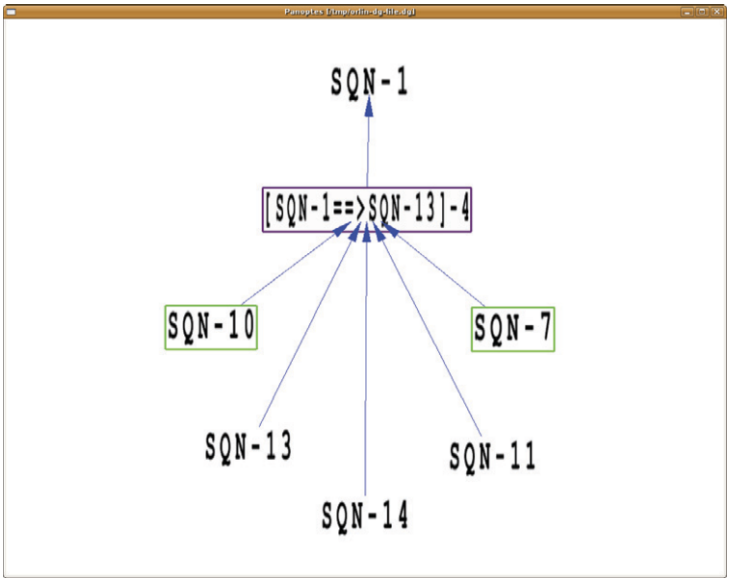
The source code and instructions for compiling and running the system are available at the Panoptes home page: http://imps.mcmaster.ca/ogrigorov/panoptes/. The home page also provides access to a demo of the system, as well as detailed documentation of the requirements, design, and implementation of the system [8].

A few screenshots are displayed below, although the complete functionality, features, and performance of Panoptes cannot be demonstrated by static pictures:

- **Screenshot 1.** IMPS and Panoptes working side by side.

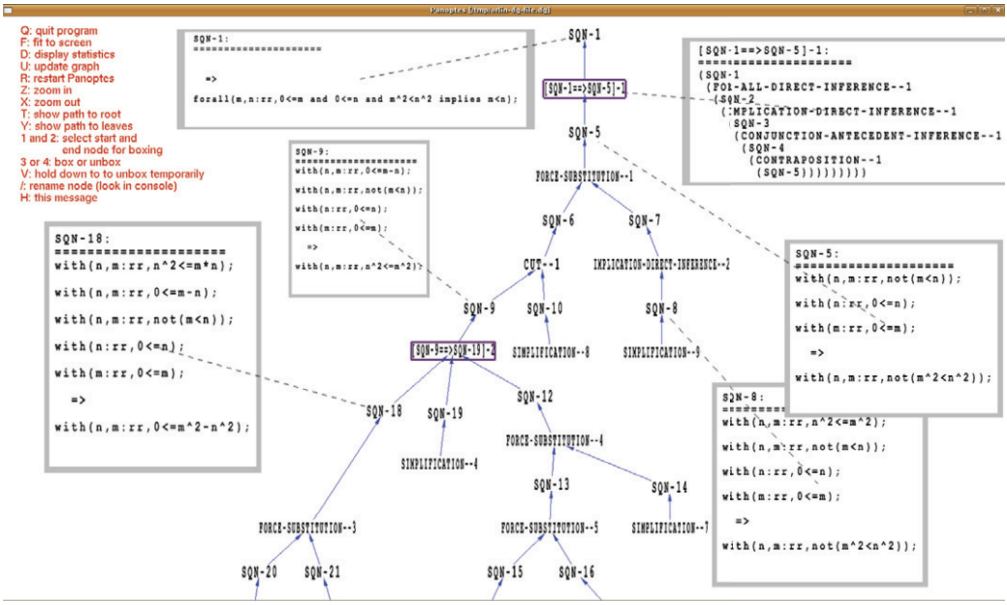


- **Screenshot 2.** The deduction graph from the previous screenshot is fully collapsed.

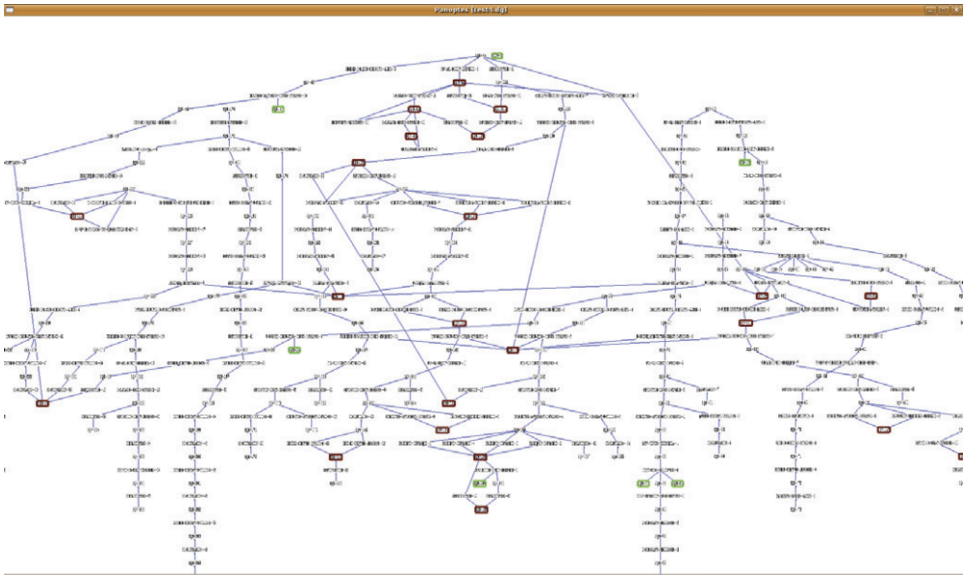


- **Screenshot 3.** A rendering of a larger graph. A help screen with the

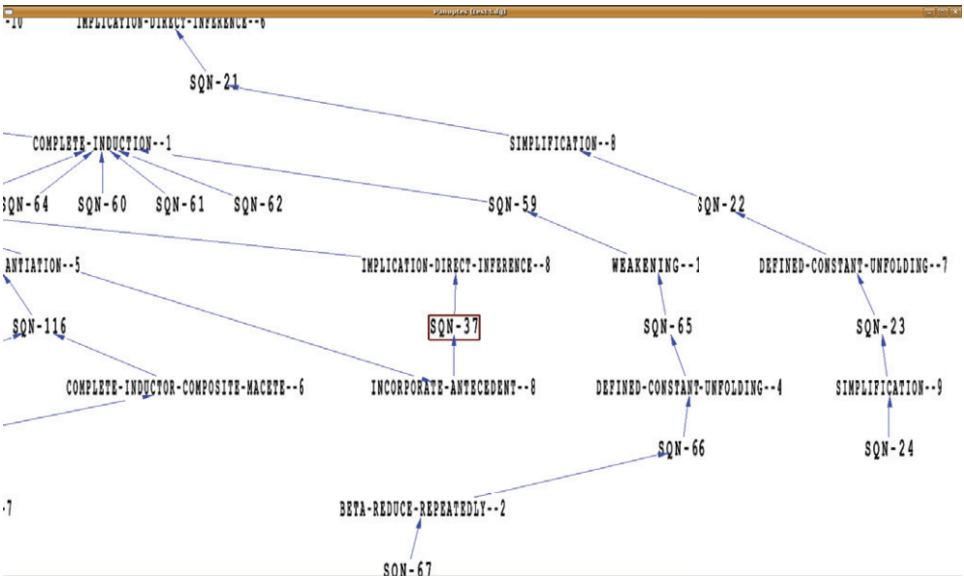
available commands is visible, and a few subgraphs are collapsed. Some information boxes for certain nodes are visible.



- **Screenshot 4.** A very large graph consisting of hundreds of nodes.



- **Screenshot 5.** Zoomed section of the large graph from the previous screenshot.



4 Related Work

A number of people have invested time and effort in improving the user experience with theorem provers. The work that deals with matter similar to the ideas behind Panoptes is described below.

Developed in Java, the *Interactive Symbolic Visualization of Semi-Automatic Theorem Proving* system [1] presents formal proofs produced by the ACL2 theorem prover [10] in the form of cone-shaped three-dimensional graphs on the screen. The user can rotate the visualization in order to look at all angles, as well as to open detached information windows with information about the nodes. Rather than labeling, colors are used to differentiate between different nodes.

Another proof assistant, PVS [11], offers graphical display of proof trees for users with Tcl/Tk (http://www.tcl.tk/) installed on their systems. The visualization appears to have limited functionality for the manipulation of the tree display, although it too offers opening of windows with details about the nodes.

The Interactive Derivation Viewer [15] renders derivations that are written in the TPTP [14] language [13], and provides an interface that allows one to quickly explore different features of the derivation. The display is very customizable and the user has access to functions like zooming, fit to height or width of the drawing pane, etc. The user can also see the information asso-

ciated with each node, although it appears difficult to have such information visible simultaneously for more than one node.

LOUI (Lovely ΩMEGA User Interface) [12] offers a graphical representation of the logical proofs created by the ΩMEGA system [2] in the form of a graphical structure that is a proper tree. It divides the nodes into different categories and differentiates them visually from one another by assigning different shapes and colors to each category. Similarly to Panoptes, the user has different facilities to manipulate the appearance of the proof tree, such as zooming, scrolling, focusing, and other functionalities.

5 Future Work

Future work can take different directions.

New features, such as a facility to syntactically or semantically compare and calculate a degree of similarity between sequent nodes will further enhance the effectiveness of Panoptes. Also, exploiting the 3D capabilities provided by OpenGL can result in the ability to stack different proof attempts of a particular goal perpendicularly to the screen plane. The user can then use commands to spin through the different proof attempts or even look at the graph from a different angle for obtaining different perspective and understanding.

Even though Panoptes was successfully tested and performed without noticeable lags on a system equipped with two 30" Apple Cinema HD™ displays, each capable of 2560×1600 pixels resolution, it is yet to be tested on a system connected to a large wall of screens (i.e., 4×3 units with combined resolution of 10,240×4,800 pixels). Since the current design and implementation concentrate on optimizing the program for better runtime performance, it will prove beneficial if the tool is running smoothly on such large screen systems.

Since the dataflow between IMPS and Panoptes is currently happening only in one direction (data can travel from IMPS to Panoptes, but Panoptes cannot send messages to IMPS), expanding Panoptes into a standalone user interface to completely replace the existing Emacs-based user interface of IMPS is the most ambitious plan of all. This is due to the enormous amount of details that need to be accounted for, although given sufficient time and dedication, it is completely achievable.

6 Conclusion

The users of proof assistants require effective tools for exploring the proof structures they create. Panoptes demonstrates the kind of functionality that these tools need to provide. Its implementation utilizes the powerful features

offered by today's computer graphics technology. The ideas used in Panoptes for exploring IMPS deduction graphs can be readily applied to other proof assistants. Moreover, Panoptes has been designed so that the code itself can be ported to other proof assistants as well.

References

[1] Bajaj, C., S. Khandelwal, J. Moore and V. Siddavanahalli, *Interactive symbolic visualization of semi-automatic theorem proving*, Technical Report TR-03-37, University of Texas at Austin (2003).

[2] Benzmller, C., L. Cheikhrouhou, D. Fehrer, A. Fiedler, Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann and V. Sorge, *Omega: Towards a mathematical assistant*, in: W. McCune, editor, *Automated Deduction—CADE-14*, Lecture Notes in Computer Science **1249** (1997), pp. 252–255.

[3] Farmer, W. M., J. D. Guttman and F. J. Thayer, *IMPS: An Interactive Mathematical Proof System*, Journal of Automated Reasoning **11** (1993), pp. 213–248.

[4] Farmer, W. M., J. D. Guttman and F. J. Thayer, *The IMPS user's manual*, Technical Report M-93B138, The MITRE Corporation (1993), online at http://imps.mcmaster.ca/.

[5] Farmer, W. M., J. D. Guttman and F. J. Thayer Fábrega, *IMPS: An updated system description*, in: M. McRobbie and J. Slaney, editors, *Automated Deduction—CADE-13*, Lecture Notes in Computer Science **1104** (1996), pp. 298–302.

[6] Garrigue, J., *An Objective Caml interface to OpenGL* (2007), online at http://wwwfun.kurims.kyoto-u.ac.jp/soft/olabl/lablgl.html.

[7] Gold Standard Group and SGI, *OpenGL—The industry standard for high performance graphics*.

[8] Grigorov, O. G., "Panoptes: An Exploration Tool for Formal Proofs," M.Sc. in Computer Science thesis, McMaster University (2008), online at http://imps.mcmaster.ca/ogrigorov/panoptes.

[9] INRIA, *The Caml language* (2008), online at http://caml.inria.fr.

[10] Kaufmann, M. and J. Moore, *An industrial strength theorem prover for a logic based on common lisp*, Software Engineering **23** (1997), pp. 203–213.

[11] Owre, S., N. Shankar, J. M. Rushby and D. W. J. Stringer-Calvert, "PVS System Guide, Version 2.4," SRI International, Menlo Park, CA, USA, November 2001.

[12] Siekmann, J., S. Hess, C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, H. Horacek, M. Kohlhase, K. Konrad, A. Meier, E. Melis and V. Sorge, *LOUI: A distributed graphical user interface for the interactive proof system* ΩMEGA, in: R. C. Backhouse, editor, *User Interfaces for Theorem Provers*, number 98-08 in Computing Science Reports, Department of Mathematics and Computing Science, Eindhoven Technical University, 1998, pp. 130–138.

[13] Sutcliffe, G., S. Schulz, K. Claessen and A. V. Gelder, *Using the TPTP language for writing derivations and finite interpretations.*, in: U. Furbach and N. Shankar, editors, *IJCAR*, Lecture Notes in Computer Science **4130** (2006), pp. 67–81.

[14] Sutcliffe, G. and C. Suttner, *The TPTP problem library*, J of Automated Reasoning **21** (1998), pp. 177–203.

[15] Trac, S., Y. Puzis and G. Sutcliffe, *An interactive derivation viewer*, Electronic Notes in Theoretical Computer Science **174** (2007), pp. 109–123.